EE2703 Assignment 4

Author: Anand Uday Gokhale, EE17B158

23^{rd} February 2019

1 Abstract

We will fit two functions, e^x and $cos(cos(x))$ over the interval $[0, 2\pi)$ using their computed Fourier series coefficients.

2 Introduction

The Fourier Series of a function $f(x)$ with period 2π is computed as follows:

$$f(x) = a_0 + \sum_{n=1}^{+\infty} \{a_n cos(nx) + b_n sin(nx)\} \tag{1}$$

where ,

$$a_0 = \frac{1}{2\pi} \int_0^{2\pi} f(x)dx$$

$$a_n = \frac{1}{2\pi} \int_0^{2\pi} f(x) * \cos(nx)dx$$

$$b_n = \frac{1}{2\pi} \int_0^{2\pi} f(x) * \sin(nx)dx$$

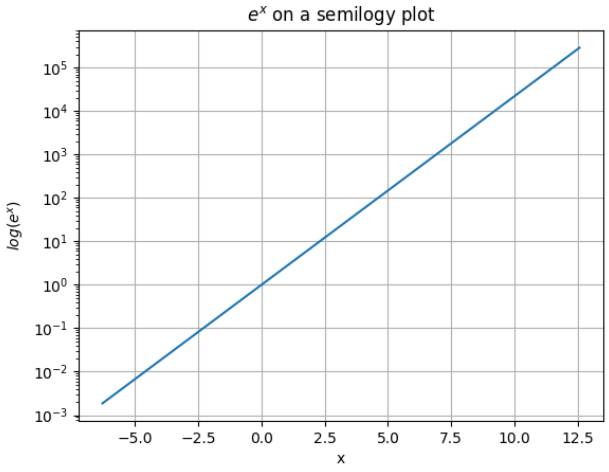
For the sake of this assignment, Since e^x doesn't have a period of 2π,We choose to change its definition in a piece-wise manner to satisfy periodicity.

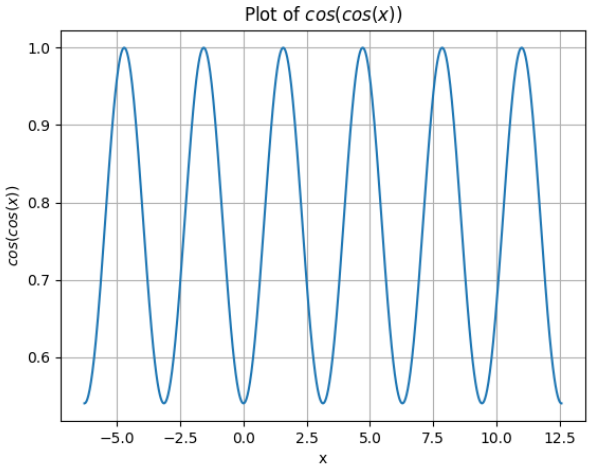
3 Assignment Questions

3.1 Creating the functions

$cos(cos(x))$ is a periodic function with period 2π whereas e^x is not. The functions that I think will be generated from the Fourier series are $cos(cos(x))$ and $e^{x\%(2\pi)}$

```
def helper(fun):
    def newfun(x):
        return fun(np.remainder(x,2*np.pi))
    return newfun
def e(x):
    return np.exp(x)
e = helper(e)
def coscos(x):
    return np.cos(np.cos(x))
```





3.2 Generating Fourier Coefficients

The first 51 coefficients are generated using the scipy.integrate.quad and the equations mentioned in the introduction function.They are saved in the following form as required by part 3:

$$\begin{bmatrix} a_0 \\ a_1 \\ b_1 \\ \dots \\ a_{25} \\ b_{25} \end{bmatrix}$$

```
def FT(n,function):
    a = np.zeros(n)
    def fcos(x,k,f):
        return f(x)*np.cos(k*x)/np.pi
    def fsin(x,k,f):
        return f(x)*np.sin(k*x)/np.pi

    a[0] = integrate.quad(function,0,2*np.pi)[0]/(2*np.pi)
    print(a[0])
    for i in range(1,n):
        if(i%2==1):
            a[i] = integrate.quad(fcos,0,2*np.pi,args=(int(i/2)+1,function))[0]
        else:
            a[i] = integrate.quad(fsin,0,2*np.pi,args=(int(i/2),function))[0]
    return a
```

3.3 Visualizing Fourier Coefficients

As expected, $\sum \|b_n\|$ for cos(cos(x)) = 2.09e-14 which is almost 0. The coefficients for e^x decay faster than that of The Log-log plot for Fourier coefficients of e^x is nearly linear because :

$$\int_0^{2\pi} e^x cos(kx)dx = \frac{(e^{2\pi}-1)}{(k^2+1)} \tag{2}$$

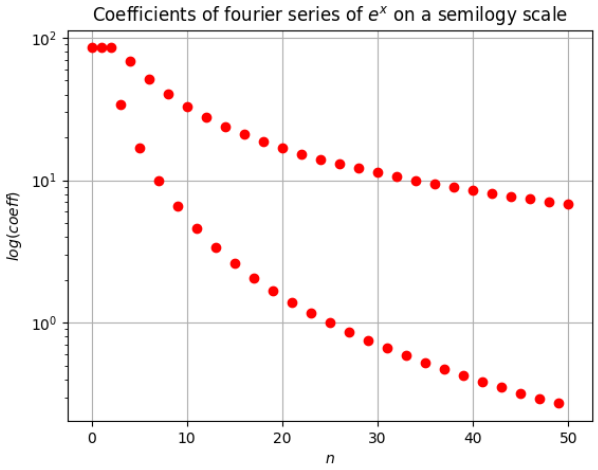
and

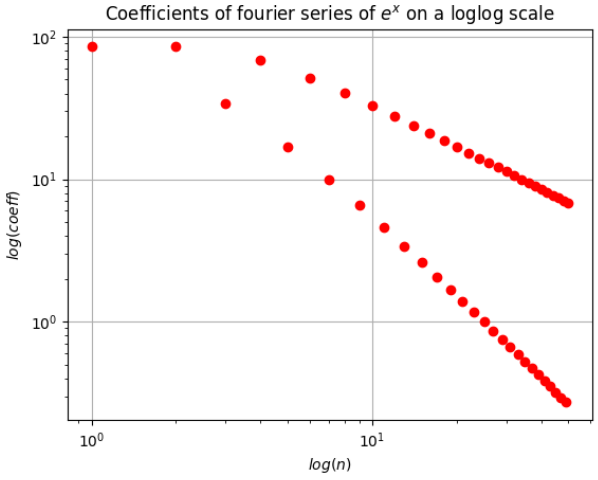
$$\int_0^{2\pi} e^x sin(kx)dx = \frac{(-ke^{2\pi}+k)}{(k^2+1)} \tag{3}$$

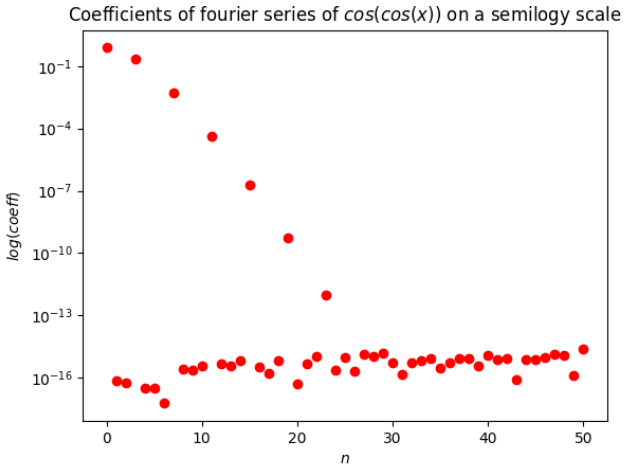
the log-log plots of these functions are linear

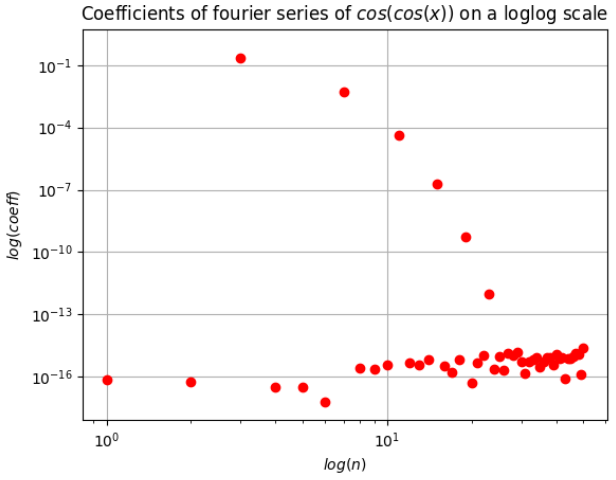
The semi-logy plot for Fourier Coefficients of $cos(cos(x))$ is linear as the integral converges to a Linear Combination of Bessel functions which are proportional to e^x.

```
def plot4(eFT,cosFT,color = 'ro'):
    eFT = np.abs(eFT)
    cosFT = np.abs(cosFT)
    plt.title(r"Coefficients of fourier series of $e^x$ on a semilogy scale")
    plt.xlabel(r'$n$')
    plt.ylabel(r'$log(coeff)$')
    plt.semilogy(eFT,color)
    plt.grid(True)
    plt.show()
    plt.title(r"Coefficients of fourier series of $e^x$ on a loglog scale")
    plt.xlabel(r'$log(n)$')
    plt.ylabel(r'$log(coeff)$')
    plt.loglog(eFT,color)
    plt.grid(True)
    plt.show()
    plt.title(r"Coefficients of fourier series of $cos(cos(x))$ on a semilogy scale")
    plt.xlabel(r'$n$')
    plt.ylabel(r'$log(coeff)$')
    plt.semilogy(cosFT,color)
    plt.show()
    plt.title(r"Coefficients of fourier series of $cos(cos(x))$ on a loglog scale")
    plt.xlabel(r'$log(n)$')
    plt.ylabel(r'$log(coeff)$')
    plt.loglog(cosFT,color)
    plt.grid(True)
    plt.show()

cosFT = FT(51,coscos)
eFT = FT(51,e)
plot4(eFT,cosFT)
```


Coefficients of fourier series of e^x on a semilogy scale
10^2
10^1
10^0
log(coeff)
0
10
20
30
40
50
n


Coefficients of fourier series of e^x on a loglog scale
10^2
10^1
10^0
log(coeff)
10^0
10^1
log(n)


Coefficients of fourier series of $cos(cos(x))$ on a semilogy scale
$log(coeff)$
10^{-1}
10^{-4}
10^{-7}
10^{-10}
10^{-13}
10^{-16}
0
10
20
30
40
50
n


Coefficients of fourier series of $cos(cos(x))$ on a loglog scale
$log(coeff)$
10^{-1}
10^{-4}
10^{-7}
10^{-10}
10^{-13}
10^{-16}
10^{0}
10^{1}
$log(n)$

3.4 A Least Squares Approach

Building on last weeks work we now try a Least Squares Approach to this problem. We linearly choose 400 values of x in the range [0,2π).It can be Noted that far better approximations were achieved when a larger value(10000) was used instead of 400. We try to solve Equation (1) By using regression on these 400 values

$$\begin{pmatrix} 1 & \cos(x_1) & \sin(x_1) & .... & \cos(25x_1) & \sin(25x_1) \\ 1 & \cos(x_2) & \sin(x_2) & .... & \cos(25x_2) & \sin(25x_2) \\ ... & ... & ... & .... & ... & ... \\ 1 & \cos(x_{400}) & \sin(x_{400}) & .... & \cos(25x_{400}) & \sin(25x_{400}) \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ b_1 \\ ... \\ a_{25} \\ b_{25} \end{pmatrix} = \begin{pmatrix} f(x_1) \\ f(x_2) \\ ... \\ f(x_{400}) \end{pmatrix}$$

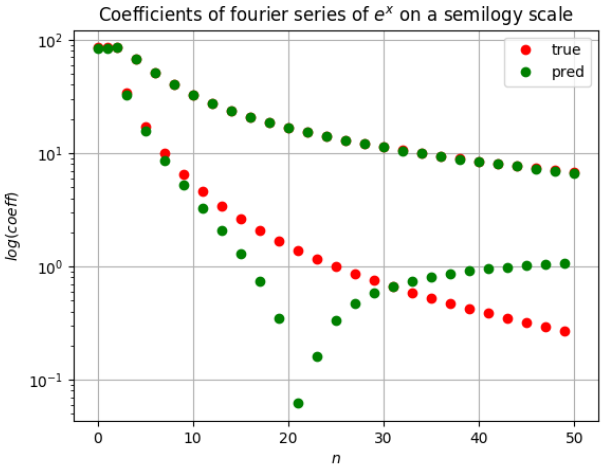
We create the matrix on the left side and call it A . We want to solve $Ac = b$ where c are the fourier coefficients.

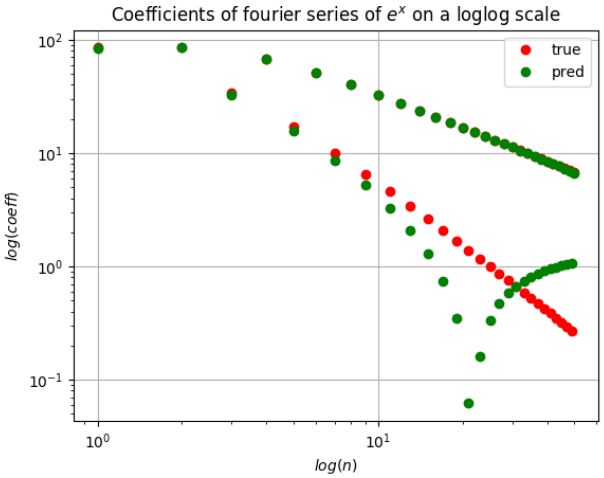
```
def generateAb(x,f):
    A = np.zeros((x.shape[0],51))
    A[:,0] = 1
    for i in range(1,26):
        A[:,2*i-1]=np.cos(i*x)

    for i in range(1,25):
        A[:,2*i]=np.sin(i*x)
    print(A.shape)
    print(f(x).shape)
    return A,f(x)
x = np.linspace(0,2*np.pi,401)[:-1]
Acos,bcos = generateAb(x,coscos)
Ae,be = generateAb(x,e)
```

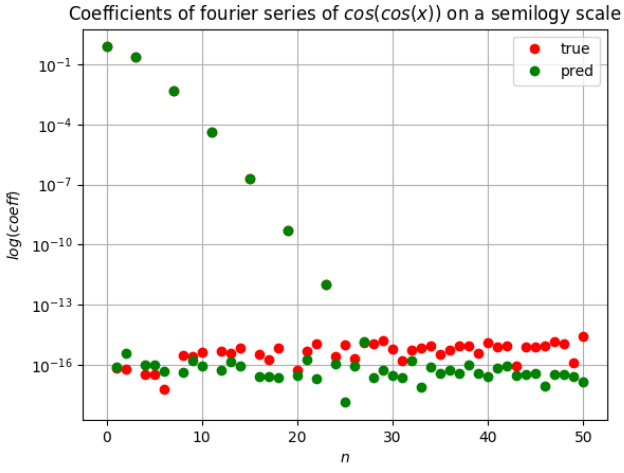
3.5 Visualizing output of the Least Squares Approach

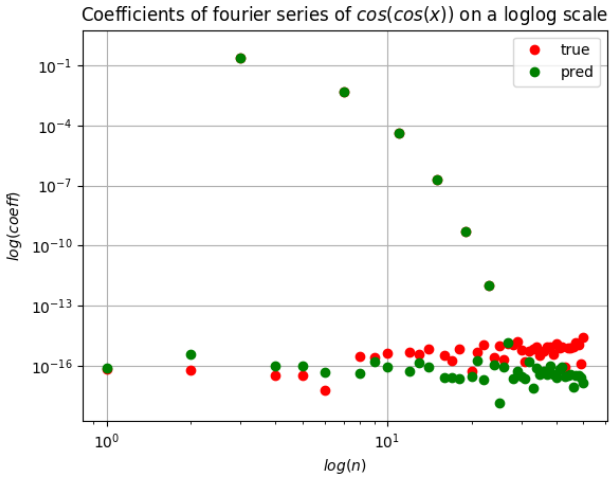
```
ccos = scipy.linalg.lstsq(Acos,bcos)[0]
ce = scipy.linalg.lstsq(Ae,be)[0]

def plot8(eFT,cosFT,ce,ccos,color = 'ro'):
    eFT = np.abs(eFT)
    cosFT = np.abs(cosFT)
    ce = np.abs(ce)
    ccos = np.abs(ccos)
    plt.title(r"Coefficients of fourier series of $e^x$ on a semilogy scale")
    plt.xlabel(r'$n$')
    plt.ylabel(r'$log(coeff)$')
    plt.semilogy(eFT,'ro')
    plt.semilogy(ce,color)
    plt.legend(["true","pred"])
    plt.grid(True)
    plt.show()
    plt.title(r"Coefficients of fourier series of $e^x$ on a loglog scale")
```

```
    plt.xlabel(r'$log(n)$')
    plt.ylabel(r'$log(coeff)$')
    plt.loglog(eFT, 'ro')
    plt.semilogy(ce, color)
    plt.legend(["true","pred"])
    plt.grid(True)
    plt.show()
    plt.title(r"Coefficients of fourier series of $cos(cos(x))$ on a semilogy scale")
    plt.xlabel(r'$n$')
    plt.ylabel(r'$log(coeff)$')
    plt.semilogy(cosFT, 'ro')
    plt.semilogy(ccos, color)
    plt.legend(["true","pred"])
    plt.grid(True)
    plt.show()
    plt.title(r"Coefficients of fourier series of $cos(cos(x))$ on a loglog scale")
    plt.xlabel(r'$log(n)$')
    plt.ylabel(r'$log(coeff)$')
    plt.loglog(cosFT, 'ro')
    plt.semilogy(ccos, color)
    plt.legend(["true","pred"])
    plt.grid(True)
    plt.show()

plot8(eFT,cosFT,ce,ccos,'go')
```




Coefficients of fourier series of e^x on a loglog scale
true
pred
$log(coeff)$
$log(n)$
10^2
10^1
10^0
10^{-1}
10^0
10^1


Coefficients of fourier series of $cos(cos(x))$ on a semilogy scale
true
pred
$log(coeff)$
n
10^{-1}
10^{-4}
10^{-7}
10^{-10}
10^{-13}
10^{-16}
0
10
20
30
40
50


Coefficients of fourier series of $cos(cos(x))$ on a loglog scale
true
pred
10^{-1}
10^{-4}
10^{-7}
10^{-10}
10^{-13}
10^{-16}
$log(coeff)$
10^{0}
10^{1}
$log(n)$

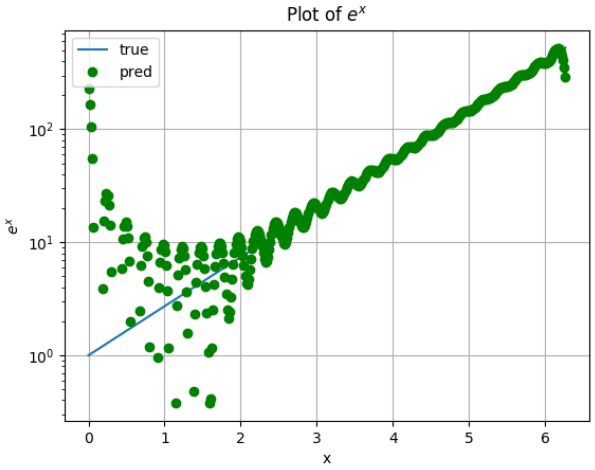
3.6 Comparing Predictions

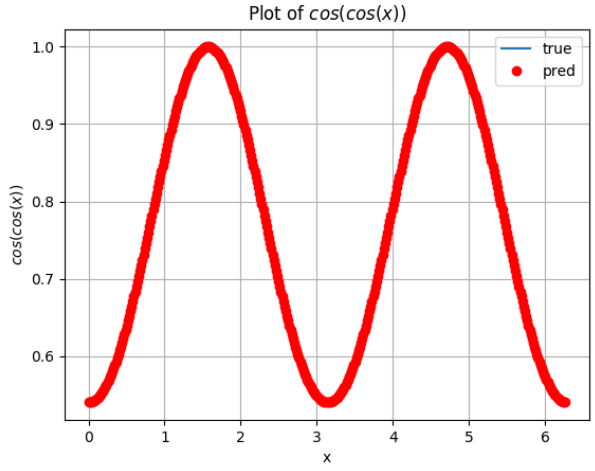
The maximum absolute error for e^x =1.3327308703353822
Whereas the same metric for cos(cos(x)) = 2.5869790664721666e-15
Our Predictions for e^x are very poor compared to that of cos(cos(x)). This can be fixed by sampling at a larger number of points.
For instance if we sample at 1e5 points :
The maximum absolute error for e^x =0.005344860583583633
Whereas the same metric for cos(cos(x)) = 2.6342502811925345e-15
This is a small change in accuracy at a very large computational cost. Hence it isn't worth it as far as this assignment is concerned.

3.7 Plotting Results

```
TTT = np.matmul(Ae,ce)
x = np.linspace(0,2*np.pi,401)[:-1]
plt.title(r"Plot of $e^x$")
plt.semilogy(x,e(x))
plt.semilogy(x,TTT,'go')
plt.xlabel('x')
plt.ylabel(r'$e^x$')
plt.legend(["true","pred"])
plt.show()

ccos = np.reshape(ccos,(51,1))
print(Acos.shape,ccos.shape)
TTT = np.matmul(Acos,ccos)

x = np.linspace(0,2*np.pi,401)[:-1]
plt.title(r"Plot of $cos(cos(x))$")
plt.plot(x,coscos(x))
plt.plot(x,TTT,'ro')
plt.xlabel('x')
plt.ylabel(r'$cos(cos(x))$')
plt.legend(["true","pred"])
plt.show()
```

It should be noted that e^x is a non periodic function and Fourier series' exists only for periodic functions. Hence we have considered a variation of e^x with period 2pi that has the actual value of e^x only in the range [0,2π). Hence it is acceptable that there is a large discrepancy in the predicted value of e^x at these boundaries


Plot of e^x
true
pred
10^2
10^1
10^0
e^x
0
1
2
3
4
5
6
x


Plot of $cos(cos(x))$
true
pred
1.0
0.9
0.8
0.7
0.6
$cos(cos(x))$
0
1
2
3
4
5
6
x

4 Conclusion

We have examined the case of approximating functions using their Fourier coefficients upto a threshold. Whilst doing so, we perform the same for two cases, one a continuous function, and the other a function with finite discontinuities.

The methods adopted in finding the respective Fourier coefficients have been direct evaluation of the Fourier series formula, as well an Least Square best fit. We notice close matching of the two methods in case of $\cos(\cos(x))$ while, there is a larger discrepancy in $\exp(x)$.

Besides this, we also highlight the fact of non-uniform convergence of the fourier series in case of finitely discontinuous functions.